Stochastic Process Algebras and their Markovian Semantics

Jane Hillston, LFCS, School of Informatics, University of Edinburgh

1. INTRODUCTION

1.1. Formalising quantitative modelling

There is a long tradition of quantitative modelling in computer and telecommunication systems for performance evaluation and dependability analysis, dating back to the early work of Erlang on telephone exchanges at the beginning of twentieth century [Erlang 1917]. These models are typically stochastic, not because the behaviour of the system under study is inherently random, but because the models usually abstract from data and because interactions with humans are often intrinsic to the system, and both of these give rise to apparently stochastic behaviour. Whilst systems remained of moderate size, models could readily be hand-crafted, and results such as Little's Law were derived and widely-used, for example to optimise job shops [Little 1961]. Much work, like Little's Law, was based on a queueing abstraction, treating the resource in the system as the server of a queue, with users following an appropriate queueing discipline before gaining access to the resource. The more contention there is for the resource, the longer jobs are likely to spend in the queue waiting for access. The queueing paradigm is based on an essentially sequential view of computation and communication, with jobs flowing between resources, completing one task after another. Nevertheless it functioned extremely well until the 1980s, and for many people *performance modelling* was synonymous with *queueing theory*. However, the arrival of distributed systems that allowed jobs to access more than one resource at once, and multicast and broadcast communication, broke the assumptions of basic queueing theory making it much harder to carry out quantitative analysis of stochastic systems in order to predict non-functional properties such as performance, availability, reliability and dependability.

It was perhaps natural that people looked to the existing formalisms for concurrency to develop high-level languages to assist in the construction of quantitative models, particularly in the area of performance modelling. At the underlying level the desire is to represent the behaviour of the system as a Continuous time Markov chain (CTMC), which can be regarded as a state machine, decorated with information about the timing and likelihood of each transition. Specifically transitions between states are assumed to be subject to a delay. This delay is governed by a random variable. Typically the uncertainty over time arises because details of the system are abstracted in the model — the size of the message to be communicated, the amount of computation to be completed on the server, etc. — and in the case of a CTMC it is assumed the delays are all governed by a negative exponential distribution. This is a choice of expediency based on the availability of analytical and numerical solution techniques that are available for CTMC models, but not for more general stochastic processes.

In the early 1990s several stochastic extensions of process algebra appeared in the literature, motivated by a desire to address the need for high-level quantitative mod-

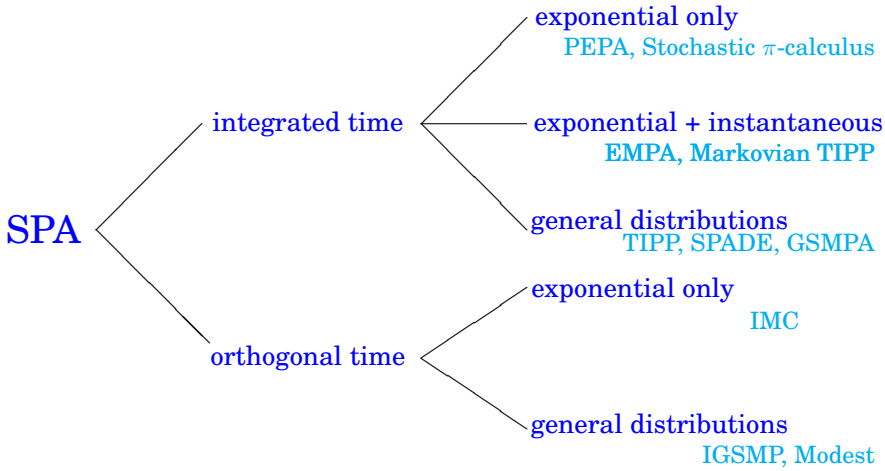


Fig. 1. Classification of the early stochastic process algebras

elling languages. These were collectively known as stochastic process algebra (SPA) or sometimes Markovian process algebras (MPA) when they were restricted to delays governed by negative exponential distributions. These included TIPP [Rettelbach and Siegle 1994] from the University of Erlangen, EMPA[1] [Bernardo et al. 1994; Bernardo and Gorrieri 1998] from the University of Bologna, PEPA [Hillston 1995] from the University of Edinburgh, MPA from the University of Dortmund [Buchholz 1994], stochastic π-calculus [Priami 1995] from the University of Pisa and SPADE[2] [Strulo 1993] from Imperial College. In these languages the basic action becomes (α, λ), where α is the action type as in classical process algebras and λ is the parameter of the associated exponentially distributed delay. PEPA was the first language to be specifically developed with the intention of generating CTMCs which could be solved numerically for performance evaluation, but subsequently versions of TIPP and EMPA were also given a Markovian semantics. Other Markovian-based SPAs, such as IMC [Hermanns 2002], emerged later. Often what distinguishes the languages is how synchronisation, particularly between timed actions, is treated. In particular IMC separates actions α, which are all assumed to be instantaneous, from the progression of time (denoted (λ)), meaning that there are two distinct transition relations in the semantics, one for logical progress and one for the passage of time.

Not all SPA have been restricted to the Markovian case: SPADE was given a semantics in terms of generalised semi-Markov processes, to be solved by simulation. Subsequently several other calculi have also incorporated generally distributed activities or delays, e.g. Modest [D'Argenio et al. 2001], IGSMP [Bravetti and Gorrieri 2002] and GSMPA [Bravetti 2002], generally giving rise to a semantics in terms of generalised semi-Markov processes. These formalisms retain the compositional structure of classical process algebras and the additional information captured within the model allows analysis to investigate additional properties such as dynamic behaviour and resource usage.

[1]Originally called simply MPA.
[2]Originally called CCS+.

1.2. Widening areas of application

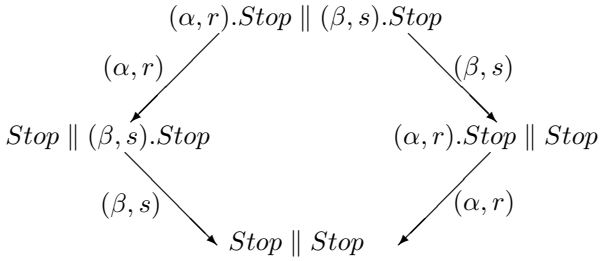
In a highly influential paper in 2002, Regev and Shapiro pointed out an analogy between communicating concurrent processes and intra-cellular biological processes [Regev and Shapiro 2002]. They suggested that formal languages such as process algebras could be used to describe such processes and this led to a plethora of adopted and developed process algebras for describing cellular processes e.g. [Cardelli 2005; Priami 1995; Regev et al. 2004; Ciocchetta and Hillston 2009], as well as related formalisms such as Kappa [Krivine 2017]. Meanwhile the process-based approach to building quantitative models, supported by stochastic process algebras was adopted in a number of other fields including ecology [Philippou et al. 2013; Vissat et al. 2016], [Massink and Latella 2012], human movement [Massink et al. 2012; Galpin et al. 2018] and collective adaptive systems [Loreti and Hillston 2016; Massink et al. 2013].

Whilst most of this work focussed on the discrete state semantics and subsequent discrete event simulation of the underlying CTMC using Gillespie's algorithm [Gillespie and Petzold 2006], work such as [Calder et al. 2005; Cardelli 2008a] established that it was also possible to derive the systems of ODEs more familiar to biologists from process algebra descriptions.

1.3. Interplay between the modelling formalism and the underlying mathematical model

The original motivation for the development of SPAs was not just to obtain compositional high-level modelling formalisms with a mathematical semantics. Especially in the area of performance analysis, there is a wealth of results on efficient and approximate approaches to extracting performance measures from CTMCs. However, many of these rely on an expert recognising a particular structure within the CTMC. Linking the CTMC to a high-level semantics opened the possibility to automatically detect and apply these approaches. Moreover, rather than having to prove on a case-by-case basis that an approximation is valid for a particular model, this can be done within the semantics of the language, establishing the validity of the approximation for all conforming models.

The memoryless property of the negative exponential distribution, and consequently CTMCs, made CTMC a good semantic target for SPAs. Usually portrayed as the condition that future behaviour is independent of all history except the current state, the memoryless property also means that once within a state, a CTMC "forgets" how long it has been there: a transition is as likely to happen at any instant of time regardless of how long the process has been in a state. From the process algebra perspective, this allows us to recover the well-known *expansion law* which underlies the interleaving semantics [Milner 1980]. Thus if we consider a process $(\alpha, r).Stop \parallel (\beta, s).Stop$, from the semantics we derive:



In a generally timed (or even deterministically timed) scenario it would be important to record the elapsed time in the intermediate states in order to know the residual time of the remaining activity. For example, the time needed to complete β in $Stop \parallel (\beta, s).Stop$ should reflect the time already taken to complete activity α. However the

memoryless property of the exponential distribution tells us that the distribution of the residual time in β is the same as it was initially in state $(\alpha, r).Stop \parallel (\beta, s).Stop$ before any time had elapsed. Thus we retain the expansion law of classical process algebra:

$$(\alpha, r).Stop \parallel (\beta, s).Stop = (\alpha, r).(\beta, s).(Stop \parallel Stop) + (\beta, s).(\alpha, r).(Stop \parallel Stop)$$

Later formalisms that incorporated general distributions either avoided the issue of residual durations by separating actions and delays (e.g. Modest [D'Argenio et al. 2001] and IGSMP [Bravetti and Gorrieri 2002]), or used a finer-grained semantics such as ST-semantics to distinguish the start and stop of each action (e.g. GSMPA [Bravetti 2002]).

There has been much work exploring how the properties of the high-level languages can be exploited to assist in the analysis of the underlying model through a variety of techniques (e.g. decomposition [Ciardo and Trivedi 1993; Hillston 2000], aggregation based on bisimulation [Hillston 1995], etc).

2. DISCRETE SEMANTICS

In all these models the entities in the system under study are represented as components in the process algebra. The structured operational semantics of the language is used to identify all possible behaviours of the system as a labelled transition system. As the example presented above suggests, it is straightforward to treat the "(α, λ)" action term as the label in the labelled transition system when the negative exponential distribution is used. Moreover this form of distribution for delays ensures that the labelled transition system can be interpreted as a CTMC, providing access to a wide array of analysis techniques, usually in terms of the evolution of the probability distribution over states of the model over time. In the following we exemplify the derivation of the discrete semantics for the SPA PEPA [Hillston 1995].

2.1. PEPA

PEPA is a CSP-like process calculus extended with the notion of exponentially distributed activities. A PEPA model consists of a collection of components which undertake actions. A component may perform an action autonomously (*independent actions*) or in synchronisation with other components in the system (*shared actions*). The language supports the following operators:

Prefix $(\alpha, r).E.$ denotes a designated first activity, with type α and activity rate r, where $r \in \mathbb{R}_{>0} \cup \{\top\}$, and $\top$ denotes a form of *passive synchronisation*.
Choice $E + F.$ indicates a stochastic choice: the choice is resolved by a *race condition* between E and F. For instance, let $r, s > 0$, in the choice $(\alpha, r).E + (\beta, s).F$ the actions α and β are executed with probabilities $r/(r + s)$ and $s/(r + s)$, respectively.
Constant $A \stackrel{\text{def}}{=} E.$ is used to model cyclic behaviour.
Cooperation $E \underset{L}{\bowtie} F.$ is the synchronisation operator of PEPA. The components E and F are required to synchronise over the action types in the set L. All the other actions are performed autonomously.
Hiding $E/L.$ relabels the activities of E with the *silent action* τ for all types in L.

PEPA models are generally taken to be those which can be generated by the following two-level grammar:

$$S ::= (\alpha, r).S \quad | \quad S + S \quad | \quad A_S, \quad A_S \stackrel{def}{=} S$$
$$C ::= S \quad | \quad C \underset{L}{\bowtie} C \quad | \quad C/L \quad | \quad A_C, \quad A_C \stackrel{def}{=} C$$

$$Client \stackrel{def}{=} (comm, r_d).Think$$
$$Think \stackrel{def}{=} (think, r_t).Client$$
$$Server \stackrel{def}{=} (comm, r_u).Log$$
$$Log \stackrel{def}{=} (log, r_l).Server$$
$$System_1 \stackrel{def}{=} Client[N_C] \underset{\{comm\}}{\bowtie} Server[N_S]$$

Fig. 2. Simple PEPA model of a client/server scenario.

The first production defines *sequential components*, i.e., processes which only exhibit sequential behaviour (by means of the prefix operator), and branching (by means of the choice operator). The second production defines *model components*, in which the interactions between the sequential components are expressed through the cooperation and hiding operators. The *system equation* designates the model component that defines the environment which embraces all of the behaviour of the system under study.

The model shown in Figure 2 is defined using such a grammar and will be used in the following to illustrate the main properties of PEPA. The model may represent a basic client/server scenario. A client is a sequential component which cycles between the two derivatives $Client$ and $Think$. Similarly, a server is a two-derivative component with derivatives $Server$ and Log. In derivative $Client$ the client is able to carry out a shared action $comm$ in cooperation with the server's derivative $Server$. The derivatives $Think$ and Log model autonomous activities performed by the components. In a distributed application, activities of this kind may be used to denote genuinely local computations or to abstract away interactions with other components with negligible impact on the performance characteristics of the system. The system equation includes the derived syntax of a *component array* $E[N]$ which occurs frequently in the modelling of large-scale systems representing a population of N independent and identical sequential components E. More formally, $E[N]$ is a shorthand notation for $\underbrace{E \underset{\emptyset}{\bowtie} E \cdots \underset{\emptyset}{\bowtie} E}_{N}$.

The set of semantic rules, shown in Figure 3, detail the possible evolutions of a term. Transitions are labelled by the activities and thus contain information about the dynamic behaviour in terms of the expected rate of the transition in addition to the type of activity performed. This inclusion of information about the rates within the labelled transition system means that a multi-transition system must be used in order to correctly reflect the dynamics of the system, i.e. if there are multiple instances of the same transition the resulting action will occur at a faster rate than if there is only a single instance, because each instance contributes to the apparent rate of the action. Thus there is not a rule of the form $(\alpha, r).P \equiv (\alpha, r).P + (\alpha, r).P$ analogous to the CCS rule $\alpha.P \equiv \alpha.P + \alpha.P$.

Most of the rules are straightforward, and presented here without comment. Rule C_2 is the fundamental inference for the characterisation of the dynamic behaviour of a shared action. It implements the semantics of *bounded capacity*: informally, the overall rate of execution of a shared activity is the minimum between the rates of the synchronising components. The rule relies on the notion of *apparent rate* to compute the total capacity of a cooperating component, according to the following definition.

Prefix

$$\mathrm{S}_0 : \frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\mathrm{S}_1 : \frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E' + F} \qquad \mathrm{S}_2 : \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} E + F'}$$

Cooperation

$$\mathrm{C}_0 : \frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F}, \alpha \notin L \qquad \mathrm{C}_1 : \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'}, \alpha \notin L$$

$$\mathrm{C}_2 : \frac{E \xrightarrow{(\alpha,r_1)} E' \qquad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'}, \alpha \in L \qquad where \qquad R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min\left(r_\alpha(E), r_\alpha(F)\right)$$

Hiding

$$\mathrm{H}_0 : \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L}, \alpha \notin L \qquad \mathrm{H}_1 : \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L}, \alpha \in L$$

Constant

$$\mathrm{A}_0 : \frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'}, A \stackrel{def}{=} E$$

Fig. 3. Markovian semantics of PEPA.

The *apparent rate* of action α in process E, denoted by $r_\alpha(E)$, indicates the overall rate at which α can be performed by E. It is recursively defined as follows:

$$r_\alpha\left((\beta, r).E\right) = \begin{cases} r & \text{if} \quad \beta = \alpha \\ 0 & \text{if} \quad \beta \neq \alpha \end{cases}$$

$$r_\alpha(E + F) = r_\alpha(E) + r_\alpha(F)$$

$$r_\alpha\left(E \underset{L}{\bowtie} F\right) = \begin{cases} \min\left(r_\alpha(E), r_\alpha(F)\right) & \text{if } \alpha \in L \\ r_\alpha(E) + r_\alpha(F) & \text{if } \alpha \notin L \end{cases}$$

$$r_\alpha(E/L) = \begin{cases} r_\alpha(E) & \text{if} \quad \alpha \notin L \\ 0 & \text{if} \quad \alpha \in L \end{cases}$$

According to this definition, for the array of sequential components $Client[N_C]$ the apparent rate of $comm$ is

$$r_{comm}\left(Client[N_C]\right) = N_C\, r_{comm}\left(Client\right) = N_C \times r_d. \tag{1}$$

Similarly,

$$r_{comm}\left(Server[N_S]\right) = N_S\, r_{comm}\left(Server\right) = N_S \times r_u. \tag{2}$$

Once the labelled transition system, or *derivation graph*, corresponding to a PEPA model has been constructed then it can be interpreted as the state transition diagram of a CTMC. In this CTMC each state corresponds to a distinct syntactic form of the PEPA expression, as the model evolves according to the semantics. The CTMC is stored as an infinitesimal generator matrix, a matrix which captures the rates of transitions between states. From this the probability distribution over the states of the model at any given time, or at steady state, can be readily derived using standard linear algebra algorithms.

Other SPA languages with integrated time have similar semantics, the major differences arising with respect to the treatment of synchronised or shared actions. In TIPP a product of rates is used, whereas in EMPA only one active partner is allowed in each synchronisation, with all other components being passive with respect to that action type. A detailed discussion of these differences is presented in [Hillston 1994].

2.2. Equivalence Relations and Model Manipulation

The state space underlying a SPA model can rapidly grow to exceed the capability of many solution techniques (operating at the level of the underlying CTMC). Thus there has been much research into how model simplification and aggregation techniques can be applied in the process algebra setting. Many such techniques are known in the context of CTMCs but are based on conditions phrased in terms of the structure of the Markov process or its generator matrix. Moreover application of these techniques often relies on the expertise of the modeller. The challenge for SPA has been to define such model manipulation techniques in the context of the process algebra, in so that they can subsequently be applied automatically. Such results have focused on the use of equivalence relations to provide the basis for comparing and manipulating models, with added benefits when the relation is a congruence with respect to the combinators of the language. The principal result is the correspondence between a Markovian form of strong bisimulation and a known concept from CTMC theory called *lumpability*. Arbitrary aggregations of a CTMC, to form macro-states by lumping together related states, in general break the memoryless property and result in a stochastic process that is no longer a CTMC. However if the aggregation is conducted based on a *strongly lumpable* partition of the state space, the resulting stochastic process constructed from the macro-states is again a CTMC. In [Hillston 1995] it was shown that Markovian bisimulation gives rise to a strongly lumpable partition; moreover the relation is a congruence with respect to the PEPA combinators, meaning that the aggregation can be carried out compositionally, as the model is constructed.

3. CONTINUOUS SEMANTICS

The discrete semantics have the advantage of providing a fine-grained view of the system, allowing the quantitative characteristics of individual entities to be derived. Unfortunately it has the disadvantage that generation and manipulation of the necessary CTMC can be very computationally expensive, or even intractable, due to the well-known *state space explosion* problem. This problem becomes particularly acute in situations where there are large numbers of entities exhibiting similar behaviour interacting within a system. Often in these situations whilst it is important to capture

the behaviour of individual entities accurately, the dynamics of the system are most fruitfully considered at a population level. Examples include the spread of information in a network, the behaviour of crowds and swarms or scalability studies involving a large number of users trying to access a service. In these cases we are interested in the *collective* rather than the *individual* dynamics.

In order to tackle this problem, a new approach to the analysis of SPA models that offers a way to avoid this state space explosion problem, at least for one class of models. When the system under consideration can be presented as a *population* model and the populations involved are known to be *large*, then a good approximation of the discrete behaviour can be achieved through a *continuous* or *fluid* approximation. Moreover, this model is scale-free in the sense that the computational effort to solve it remains the same even as the populations involved grow larger. Of course, there is a cost, in the sense that some information is lost and it is no longer possible to analyse the system in terms of individual behaviours. But when average behaviours or expectations are required, for example in situations of collective behaviour, the fluid approach has substantial benefits. At heart, this is a fluid approximation, as highlighted by Kurtz [Kurtz 1981].

3.1. Kurtz's Theorem

Kurtz's result, which dates back to the 1970s [Kurtz 1970], establishes that a sequence of CTMCs which satisfy some conditions and represent, essentially, the same system under growing populations, converges to a set of ordinary differential equations (ODEs). At convergence the behaviour of the CTMC (with infinite population) is indistinguishable from the behaviour of the set of ODEs. This theoretical limit is at an infinite population. Nevertheless in many practical cases we find empirically that sufficient convergence is often achieved at much lower populations.

Considering, for example a PEPA model with many instances of the components within the model, we can assume that there is a single agent type which is a finite state machine, with internal states taken from a finite set S, and labelled by integers: $S = \{1, 2, \ldots, n\}$. Where we have components with different behaviours we assume they have different starting states within the finite state machine, and that the reachable states for each starting state are distinct. In this way we can reason in terms of a single population of agents, even though they may exhibit distinct behaviours. We have a population of N agents, and denote the state of agent i at time t, for $i = 1, \ldots, N$, by $Y_i^{(N)}(t) \in S$. Note that we have made explicit the dependence on N, the total population size.

A configuration of a system is thus represented by the tuple $(Y_1^{(N)}, \ldots, Y_N^{(N)})$. This representation is based on treating each agent as a distinct individual with identity conferred by the position in the vector. Alternatively, assuming that agents in the same internal state cannot be distinguished, we can shift to the population view, sometimes termed a *counting abstraction*:

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}, \tag{3}$$

where $\mathbf{1}\{Y_i^{(N)} = j\}$ is an indicator function with value 1 when $Y_i^{(N)} = j$ and zero, otherwise. Note that the vector $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has a dimension independent of N; it is referred to as the population vector. The domain of each variable $X_j^{(N)}$ is $\{0, \ldots, N\}$, and if we assume components do not enter or leave the system, it holds

that $\sum_{j=1}^{n} X_j^{(N)} = N$. Let us denote with $\mathcal{S}^{(N)}$ the subset of vectors of $\{1, \ldots, N\}^n$ that satisfy this constraint.

For example, returning to the simple PEPA model presented in Figure 2, we have two states each for the clients and the servers, giving us four states in total ($Client$, $Think$, $Server$ and Log). Moreover if we assume that there is always one server for every 99 clients then with a population of $100N$ the possible states of the system will be (n_1, n_2, n_3, n_4) where $n_1 + n_2 = 99N$ and $n_3 + n_4 = N$, assuming we start in a state such as $(99N, 0, N, 0)$. Then, allowing $N \to \infty$, we have a family of models in which the ratio of clients to servers remains constant but the total population tends to infinity.

The dynamics of the population models is expressed in terms of a set of possible *events* or *transitions*. Events are stochastic, and take an exponentially distributed time to happen. Moreover their rate may depend on the current global state of the system. Hence, each event will be specified by a rate function, and by a set of update rules, specifying the impact of the event on the population vector.

Thus we define a population model $\mathcal{X}^{(N)} = (\mathbf{X}^{(N)}, \mathcal{T}^{(N)}, \mathbf{x}_0^{(N)})$, where $\mathcal{T}^{(N)}$ is the set of transitions and $\mathbf{x}_0^{(N)}$ is the initial state. Transitions are defined in terms of an update vector to be added to $\mathbf{X}^{(N)}$ whenever the transition τ occurs $\mathbf{v}_\tau$, and a rate function $r_\tau^{(N)}(\mathbf{X})$. In [Tribastone et al. 2012] a semantics for PEPA in terms of generating functions that gives rise to such as population model $\mathcal{X}^{(N)} = (\mathbf{X}^{(N)}, \mathcal{T}^{(N)}, \mathbf{x}_0^{(N)})$ is presented. Given such a model, it is straightforward to construct the associated CTMC $\mathbf{X}^{(N)}(t)$; its state space is $\mathcal{S}^{(N)}$, while its infinitesimal generator matrix $Q^{(N)}$ is the $|\mathcal{S}^{(N)}| \times |\mathcal{S}^{(N)}|$ matrix defined by

$$q_{\mathbf{x},\mathbf{x}'} = \sum \{r_\tau(\mathbf{x}) \mid \tau \in \mathcal{T},\ \mathbf{x}' = \mathbf{x} + \mathbf{v}_\tau\}.$$

Thus for our example above our population vector is $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ and the generating functions are:

$$\begin{aligned} f(\mathbf{x}, (-1, 1, -1, 1), comm) &= \min(r_d \mathbf{x}_1, r_u \mathbf{x}_3) \\ f(\mathbf{x}, (1, -1, 0, 0), think) &= r_t \mathbf{x}_2 \\ f(\mathbf{x}, (0, 0, 1, -1), log) &= r_l \mathbf{x}_4 \end{aligned}$$

As N tends to infinity, this sequence of CTMCs converges to a set of ODEs which can be readily constructed from the generating functions for $\mathcal{T}^{(N)}$; at finite but large population levels this set of ODEs provides an approximation of the CTMC behaviour. These differential equations can be interpreted in two different ways: they can be seen as an approximation of the average of the system (usually a first order approximation, see [Bortolussi 2008; Kampen 1992]). This is often termed a *mean field* approximation. Alternatively, they can be interpreted as an approximate description of system trajectories for large populations corresponding to a functional version of the law of large numbers. A sequence of CTMCs (which can be seen as random trajectories in $\mathbb{R}^n$) for increasing population size, converge to a deterministic trajectory, the solution of the

fluid ODE. In the case of the Client-Server example the set of ODEs is:

$$\begin{aligned}
\frac{dx_1}{dt} &= -\min(r_d x_1, r_u x_3) + r_t x_2 \\
\frac{dx_2}{dt} &= \min(r_d x_1, r_u x_3) - r_t x_2 \\
\frac{dx_3}{dt} &= -\min(r_d x_1, r_u x_3) + r_l x_4 \\
\frac{dx_4}{dt} &= \min(r_d x_1, r_u x_3) - r_l x_4
\end{aligned}$$

In order to allow models of different population sizes to be compared we normalise the populations by dividing each variable by the total population N. In this way, the normalised population variables $\hat{\mathbf{X}}^{(N)} = \frac{\mathbf{X}^{(N)}}{N}$, or population densities, will always range between 0 and 1. These normalised variables are usually referred to as the *occupancy measures*, as they represent the fraction of agents which occupy each state.

Update vectors, initial conditions, and rate functions [Bortolussi et al. 2013] must all also be appropriately scaled, resulting in $\hat{\mathcal{X}}^{(N)} = (\hat{\mathbf{X}}^{(N)}, \hat{\mathcal{T}}^{(N)}, \hat{\mathbf{X}}_{\mathbf{0}}^{(N)})$ the corresponding normalised model. We require that:

— initial conditions scale appropriately: $\hat{\mathbf{X}}_{\mathbf{0}}^{(N)} = \frac{\mathbf{X_0}^{(N)}}{N}$;
— for each transition $(\mathbf{v}_\tau, r_\tau^{(N)}(\mathbf{X}))$ of the non-normalised model, define $\hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$ to be the rate function expressed in the normalised variables (obtained from $r_\tau^{(N)}$ by a change of variables). The corresponding transition in the normalised model is $(\mathbf{v}_\tau, \hat{r}_\tau^{(N)}(\hat{\mathbf{X}}))$, with update vector equal to $\frac{1}{N}\mathbf{v}_\tau$.

We further assume, for each transition τ, that there exists a bounded and Lipschitz continuous function $f_\tau(\hat{\mathbf{X}}) : E \to \mathbb{R}^n$ on normalised variables (where E contains all domains of all $\hat{\mathcal{X}}^{(N)}$), independent of N, such that $\frac{1}{N}\hat{r}_\tau^{(N)}(\mathbf{x}) \to f_\tau(\mathbf{x})$ *uniformly* on E. We denote the state of the CTMC of the N-th non-normalised (resp. normalised) model at time t as $\mathbf{X}^{(N)}(t)$ (resp. $\hat{\mathbf{X}}^{(N)}(t)$).

3.2. Deterministic limit theorem

Consider a sequence of normalised models $\hat{\mathcal{X}}^{(N)}$ and let $\mathbf{v}_\tau$ be the (non-normalised) update vectors. The *drift* $F^{(N)}(\hat{\mathbf{X}})$ of $\hat{\mathcal{X}}$, which is formally the mean instantaneous increment of model variables in state $\hat{\mathbf{X}}$, is defined as

$$F^{(N)}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \frac{1}{N} \mathbf{v}_\tau \hat{r}_\tau^{(N)}(\hat{\mathbf{X}}) \tag{4}$$

Furthermore, let $f_\tau : E \to \mathbb{R}^n$, $\tau \in \hat{\mathcal{T}}$ be the limit rate functions of transitions of $\hat{\mathcal{X}}^{(N)}$. The *limit drift* of the model $\hat{\mathcal{X}}^{(N)}$ is therefore

$$F(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \mathbf{v}_\tau f_\tau(\hat{\mathbf{X}}), \tag{5}$$

and $F^{(N)}(\mathbf{x}) \to F(\mathbf{x})$ uniformly as $N \longrightarrow \infty$, as easily checked. The fluid ODE is

$$\frac{d\mathbf{x}}{dt} = F(\mathbf{x}), \qquad with \quad \mathbf{x}(0) = \mathbf{x_0} \in S.$$

Given that F is Lipschitz in E (since all f_τ are), this ODE has a unique solution $\mathbf{x}(t)$ in E starting from $\mathbf{x_0}$. Then, one can prove the Deterministic Approximation theorem [Kurtz 1970; Darling 2002] stated below:

THEOREM 3.1. *Let the sequence* $\hat{\mathbf{X}}^{(N)}(t)$ *of Markov processes and* $\mathbf{x}(t)$ *be defined as above, and assume that there is some point* $\mathbf{x_0} \in S$ *such that* $\hat{\mathbf{X}}^{(N)}(0) \to \mathbf{x_0}$ *in probability. Then, for any* finite *time horizon* $T < \infty$*, it holds that as* $N \longrightarrow \infty$*:*

$$\mathbb{P}\left\{ \sup_{0 \leq t \leq T} ||\hat{\mathbf{X}}^{(N)}(t) - \mathbf{x}(t)|| > \varepsilon \right\} \to 0.$$

Notice that the theorem makes assertions about the trajectories of the population counts at all finite times, but nothing about what happens at steady state, i.e. when time goes to infinity.

3.3. Stochastic process algebra with fluid interpretation

Kurtz's Theorem, or the Deterministic Approximation Theorem, has been established for many years. It has been widely used but when it is used directly from a CTMC model, it is the modeller's responsibility to prove that the model satisfies the necessary conditions for application of the theory, and moreover, to derive the corresponding ODEs. This must be done on a model-by-model basis. This approach has been used for several performance and dependability models e.g. [Bakhshi et al. 2009; 2011; Benaïm and Boudec 2008; Gast and Gaujal 2010].

Incorporating mean field or fluid approximation into formal high-level languages such as SPA gives access to a scalable analysis technique which is immune to the problem of state space explosion; indeed, a technique which increases in accuracy as the size of the model grows. Moreover, from the perspective of modellers already familiar with the mean field approach, it offers the possibility to establish the conditions for convergence at the language level via the semantics, once and for all, removing the need to fulfil the proof obligation on a model-by-model basis. Moreover the derivation of the ODEs can be automated in the implementation of the language.

Work has developed in both stochastic Petri nets, e.g. [Trivedi and Kulkarni 1993; Silva and Recalde 2002; 2004] and SPAs, e.g. [Kolesnichenko et al. 2011; Hillston 2005; Bortolussi and Policriti 2009]. In the context of SPAs, it is straightforward to see that components of the process algebra description can be regarded as agents within the CTMC model, typically occupying different partitions within the notional complete state space for agents, as explained above. When multiple instances of a component are present in the same scope within the model, these constitute a *population*.

The continuous semantics approach is only applicable to models where we have interactions of large populations (parallel compositions of large numbers of components with the same behaviour) within which each component has relatively simple behaviour rather than interactions between individuals each with complex behaviour. When this is the case we need to make the shift from a state representation based on individuals, to one based on counting. How this is handled depends on the SPA but is generally straightforward. For example, in PEPA models there is a simple procedure to reduce the syntactic representation to a state vector [Hillston 2005; Tribastone et al. 2012], but in languages such as Bio-PEPA the mapping is more straightforward because the language was designed to support fluid approximation [Ciocchetta and Hillston 2009]. The actions of the algebra correspond to the events in the CTMC model, and the definition of the process and its continuation via an action is the basis for the definition of the update vector.

The first work relating process algebra and mean field models can be found in the thesis of Sumpter [Sumpter 2000]. Sumpter developed models of social insects in the

discrete synchronous process algebra WSCCS [Tofts 1990]. He then heuristically derived difference equations to capture the mean field representation of the model. This work inspired the work of Norman and Shankland [Norman and Shankland 2003], in which WSCCS is used to build models of the spread of infectious diseases and difference equation representations are derived. This led on to further work with ever more rigour introduced into the relationship between the difference equation/ODE models and the process algebra descriptions from which they were derived [McCaig et al. 2008; 2009; McCaig 2008], but in later work the authors switched from using WSCCS to using PEPA and Bio-PEPA for their modelling of epidemics.

Work in systems biology stimulated more widespread interest in the relationship between process algebra description and ODE models. The first work here was the mapping given from PEPA models constructed in a particular style, representing a reagent-centric view of biological signal transductions pathways, to equivalent ODE models, by Calder *et al.* [Calder et al. 2005]. This was subsequently generalised to more arbitrary PEPA models with large populations, where the mapping to the ODE was made completely systematic, based on an intermediate structure termed the *activity matrix* [Hillston 2005]. In the work of Bortolussi and Policriti the authors consider a different style of process algebra, stochastic Concurrent Constraint Programming (sCCP), and demonstrate a mapping, both from process algebra to ODEs and from ODEs to process algebra descriptions [Bortolussi and Policriti 2009]. At around the same time Cardelli also constructed a systematic mapping from process algebra (in this case a variant of CCS) to ODEs, using a Chemical Parametric Form as an intermediary in this case [Cardelli 2008a]. The relationship between this interpretation of the process algebra model and the discrete-state stochastic semantics is explored in [Cardelli 2008b].

These initial explorations were followed by more rigorous treatment establishing the convergence results in terms of the semantics of the SPAs. For example in [Geisweiller et al. 2008], Geisweiller *et al.*, working with a generalised form of PEPA models which allow two forms of synchronisation — both the usual PEPA synchronisation based on the bounded capacity, and the biological notion of mass action — show that the syntactically derived ODE models are indeed those which are obtained by the application of Kurtz's Theoreom, guaranteeing convergence in the limit. In [Tribastone et al. 2012], Tribastone *et al.* show how it is possible to fully formalise the derivation of the ODEs for PEPA models, via a structured operational semantics. In [Bortolussi and Policriti 2009] Bortolussi and Policriti construct a process algebra that matches a given set of ODEs in the limit. An alternative approach to the derivation of the fluid approximation model is taken in the work on Kappa [Danos et al. 2010], where the ODEs are derived as an abstract interpretation.

Some authors also considered how to make the derivation of ODEs from process algebra descriptions easier. As previously mentioned, the PEPA variant, Bio-PEPA [Ciocchetta and Hillston 2009] was explicitly constructed to maintain a counting abstraction, initially making the derivation of the activity matrix easier and later supporting a semantics in the style of [Tribastone et al. 2012]. Hayden and Bradley developed another variant of PEPA, termed Grouped PEPA, which makes clearer the population structures within models [Hayden and Bradley 2010].

4. MODELS WITH UNCERTAINTY

The output from a quantitative model is strongly dependent on the parameter values used in the model construction. In the case of SPAs these parameters are the rates that govern the timing and probability of activities. However, particularly in some application domains, it is difficult to know all the parameter values when the model is constructed. A recent development has been the development ProPPA, a SPA in which

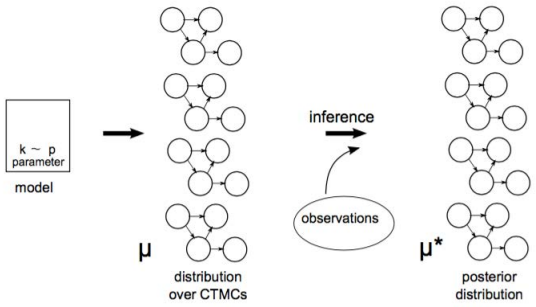


Fig. 4. Schematic view of the ProPPA modelling framework

some rates are left uncertain [Georgoulas et al. 2014]. This variant of Bio-PEPA allows some rates to be assigned a prior distribution, capturing the modeller's belief about the likely values of the rates. Moreover a model may have associated observations, collected from the system to be modelled. Thus rather than a single CTMC, the semantics of a ProPPA model is a probability distribution over CTMCs reflecting the prior distributions specified for each rate. Using inference, the model may be combined with the observations to derive updated probability distributions, over the CTMCs and over rates [Georgoulas et al. 2017]. Currently observations may be collected either as time series values of population values in a model, or as temporal logic formulas that express constraints on the observed behaviours [Georgoulas et al. 2018].

In this setting the semantic object corresponding to the SPA model is no longer a single CTMC, but a *Probabilistic Constraint Markov Chain* [Georgoulas et al. 2018]:

Definition 4.1. A *Probabilistic Constraint Markov Chain* (PCMC) is a tuple $\langle S, o, A, V, \varphi \rangle$, where:

— S is the set of states, of cardinality k.
— $o \in S$ is the initial state.
— A is a set of atomic propositions.
— $V : S \to 2^{2^A}$ gives a set of acceptable labellings for each state.
— $\varphi : S \times [0, \infty)^k \to [0, \infty)$ is the *constraint function*.

where $\varphi(s, \cdot)$ is a probability density function on the transition rates from state s.

The PCMC is closely related to *Constraint Markov Chains* (CMCs) introduced by Caillaud *et al.* in 2011 [Caillaud et al. 2011]. CMCs are a generalisation of Discrete Time Markov Chains, in which the transition probabilities are not concrete, but can take any value satisfying some constraints. In the case of PCMC we generalise CTMCs in an analogous way, but additionally capture information about the likelihood of different values that satisfy the constraints by associating a probability distribution with the satisfaction set. A structured operational semantics, derived in the FuTS style [De Nicola et al. 2013], is presented in [Georgoulas et al. 2018].

5. CONCLUSIONS

Stochastic process algebras offer a compositional modelling language which can be used for quantitive analysis and reasoning about the dynamic behaviour of systems. Their strength lies in the formal semantics in terms of mathematical structures that can be used to carry out quantitative analysis using established techniques, ranging from numerical solution of CTMCs to Bayesian inference over PCMCs. The modeller

is relieved of the task of constructing an appropriate mathematical model, using the high-level description to generate it automatically. Moreover conditions for model simplifications and approximations can be recast at the language level, allowing proof obligations to be conducted once and for all, rather than on a case-by-case basis.

Many new SPA languages have been developed in the last twenty years, and with the ever widening domains of applications there are fresh challenges for the quantitative analysis. For example, whilst the continuous semantics can be highly beneficial for collective systems with many interacting components, it is difficult to construct such a semantics for a language such as CARMA that incorporates broadcast communication.

Acknowledgement

This work is partially supported by the EU project QUANTICOL, 600708.

REFERENCES

R. Bakhshi, L. Cloth, W. Fokkink, and B.R. Haverkort. 2009. Mean-field analysis for the evaluation of gossip protocols. In *Proceedings of 6th Int. Conference on the Quantitative Evaluation of Systems (QEST 2009)*. 247–256.

R. Bakhshi, L. Cloth, W. Fokkink, and B.R. Haverkort. 2011. Mean-field framework for performance evaluation of push-pull gossip protocols. *Performance Evaluation* 68, 2 (2011), 157–179.

M. Benaïm and J. Le Boudec. 2008. A class of mean field interaction models for computer and communication systems. *Performance Evaluation* (2008).

M. Bernardo and R. Gorrieri. 1998. A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. *Theoretical Computer Science* 202, 1-2 (1998), 1–54.

M. Bernardo, R. Gorrieri, and L. Donatiello. 1994. *MPA: A Stochastic Process Algebra*. Technical Report UBLCS-94-10. Laboratory of Computer Science, University of Bologna.

L. Bortolussi. 2008. On the Approximation of Stochastic Concurrent Constraint Programming by Master Equation. *Electr. Notes Theor. Comput. Sci.* 220, 3 (2008), 163–180. DOI:http://dx.doi.org/10.1016/j.entcs.2008.11.025

L. Bortolussi, J. Hillston, D. Latella, and M. Massink. 2013. Continuous approximation of collective systems behaviour: a tutorial. *Performance Evaluation* (2013).

L. Bortolussi and A. Policriti. 2009. Dynamical systems and stochastic programming: To ordinary differential equations and back. *Trans. on Computational Systems Biology XI* 5750 (2009), 216–267.

M. Bravetti. 2002. *Specification and Analysis of Stochastic Real-Time Systems*. Ph.D. Dissertation. University of Bologna.

M. Bravetti and R. Gorrieri. 2002. The theory of interactive generalized semi-Markov processes. *Theor. Comput. Sci.* 282, 1 (2002), 5–32. DOI:http://dx.doi.org/10.1016/S0304-3975(01)00043-3

P. Buchholz. 1994. Compositional Analysis of a Markovian Process Algebra. In *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, U. Herzog and M. Rettelbach (Eds.).

B. Caillaud, B. Delahaye, K.G. Larsen, A. Legay, M.L. Pedersen, and A. Wasowski. 2011. Constraint Markov Chains. *Theor. Comput. Sci.* 412, 34 (2011), 4373–4404. DOI:http://dx.doi.org/10.1016/j.tcs.2011.05.010

M. Calder, S. Gilmore, and J. Hillston. 2005. Automatically deriving ODEs from process algebra models of signalling pathways. In *Proceedings of Computational Methods in Systems Biology (CMSB 2005)*. 204215.

L. Cardelli. 2005. Brane Calculi. In *Proceedings of Computational Methods in Systems Biology (CMSB 2004) (LNCS)*, Vol. 3082. Springer Verlag, 257–278.

L. Cardelli. 2008a. From Processes to ODEs by Chemistry. In *5th IFIP International Conference On Theoretical Computer Science - TCS 2008*. 261–281.

L. Cardelli. 2008b. On process rate semantics. *Theoretical Computer Science* 391, 3 (2008), 190–215.

G. Ciardo and K.S. Trivedi. 1993. A Decomposition Approach for Stochastic Reward Net Models. *Performance Evaluation* 18, 1 (1993), 37–59.

F. Ciocchetta and J. Hillston. 2009. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science* 410, 33–34 (2009), 3065–3085.

V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. 2010. Abstracting the Differential Semantics of Rule-Based Models: Exact and Automated Model Reduction. In *Proceedings of Logic in Computer Science (LICS 2010)*. 362–381.

P.R. D'Argenio, H. Hermanns, J-P. Katoen, and R. Klaren. 2001. MoDeST - A Modelling and Description Language for Stochastic Timed Systems. In *Process Algebra and Probabilistic Methods, Performance Modeling and Verification: Joint International Workshop, PAPM-PROBMIV 2001, Aachen, Germany, September 12-14, 2001, Proceedings (LNCS)*, Vol. 2165. Springer, 87–104.

R.W.R. Darling. 2002. Fluid limits of pure jump Markov processes: A practical guide. *arXiv.org* (2002).

R. De Nicola, D. Latella, M. Loreti, and M. Massink. 2013. A uniform definition of stochastic process calculi. *ACM Comput. Surv.* 46, 1 (2013), 5:1–5:35. DOI:http://dx.doi.org/10.1145/2522968.2522973

A.K. Erlang. 1917. Solution of some problems in th theory of probabilities of significance in automatic telephone exchanges. *Elektroteknikeren* 13 (1917), 5–13. (Danish) [English translation in the P.O. Electrical Engineering Journal 10, pp. 189-197, 1917–1918.].

V. Galpin, N. Zon, P. Wilsdorf, and S. Gilmore. 2018. Mesoscopic Modelling of Pedestrian Movement using CARMA and Its Tools. *ACM Trans. on Modeling and Computer Simulation (TOMACS)* 28, 2 (2018).

N. Gast and B. Gaujal. 2010. A mean field model of work stealing in large-scale systems. In *Proceedings of ACM SIGMETRICS 2010*. 13–24.

N. Geisweiller, J. Hillston, and M. Stenico. 2008. Relating continuous and discrete PEPA models of signalling pathways. *Theoretical Computer Science* 404, (1-2) (2008), 97–111.

A. Georgoulas, J. Hillston, D. Milios, and G. Sanguinetti. 2014. Probabilistic Programming Process Algebra. In *Quantitative Evaluation of Systems - 11th International Conference, QEST 2014, Florence, Italy, September 8-10, 2014. Proceedings (Lecture Notes in Computer Science)*, Vol. 8657. Springer Verlag, 249–264.

A. Georgoulas, J. Hillston, and G. Sanguinetti. 2017. Unbiased Bayesian inference for population Markov jump processes via random truncations. *Statistics and Computing* 27, 4 (2017), 991–1002. DOI:http://dx.doi.org/10.1007/s11222-016-9667-9

A. Georgoulas, J. Hillston, and G. Sanguinetti. 2018. ProPPA: Probabilistic Programming for Stochastic Dynamical Systems. *ACM Trans. Model. Comput. Simul.* 28, 1 (2018), 3:1–3:23. DOI:http://dx.doi.org/10.1145/3154392

D. Gillespie and L. Petzold. 2006. Numerical simulation for biochemical kinetics. In *System Modeling in Cellular Biology*. MIT Press, 331353.

R.A. Hayden and J.T. Bradley. 2010. A fluid analysis framework for a Markovian process algebra. *Theoretical Computer Science* 411, (22-24) (2010), 2260–2297.

H. Hermanns. 2002. *Interactive Markov Chains: The Quest for Quantified Quality*. Lecture Notes in Computer Science, Vol. 2428. Springer. DOI:http://dx.doi.org/10.1007/3-540-45804-2

J. Hillston. 1994. The Nature of Synchronisation. In *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, U. Herzog and M. Rettelbach (Eds.).

J. Hillston. 1995. *A Compositional Approach to Performance Modelling*. Cambridge University Press.

J. Hillston. 2000. Exploiting Structure in Solution: Decomposing Compositional Models. In *European Educational Forum: School on Formal Methods and Performance Analysis*. LNCS, Vol. 2090. Springer Verlag, 278–314.

J. Hillston. 2005. Fluid flow approximation of PEPA models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of SysTems, QEST 2005*. 33 – 42.

N.G. Van Kampen. 1992. *Stochastic Processes in Physics and Chemistry*. Elsevier.

A. Kolesnichenko, A. Remke, P.T. de Boer, and B.R. Haverkort. 2011. Comparison of the mean-field approach and simulation in a peer-to-peer botnet case study. In *Proceedings of 8th European Performance Engineering Workshop, EPEW 2011 (LNCS)*, Vol. 6977. Springer Verlag, 133–147.

J. Krivine. 2017. Systems biology. *SIGLOG News* 4, 3 (2017), 43–61. DOI:http://dx.doi.org/10.1145/3129173.3129182

T.G. Kurtz. 1970. Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability* 7 (1970), 49–58.

T.G. Kurtz. 1981. Approximation of population processes. *SIAM* (1981).

J.D.C. Little. 1961. A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research* 9, 3 (1961), 383–387. doi:10.1287/opre.9.3.383.JSTOR167570.

M. Loreti and J. Hillston. 2016. Modelling and Analysis of Collective Adaptive Systems with CARMA and its Tools. In *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems - 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2016, Bertinoro, Italy, June 20-24, 2016, Advanced Lectures (Lecture Notes in Computer Science)*, Vol. 9700. Springer, 83–119.

M. Massink, M. Brambilla, D. Latella, M. Dorigo, and M. Birattari. 2013. On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics. *Swarm Intelligence* 7, 2-3 (2013), 201–228. DOI:http://dx.doi.org/10.1007/s11721-013-0079-6

M. Massink and D. Latella. 2012. Fluid Analysis of Foraging Ants. In *Coordination Models and Languages - 14th International Conference, COORDINATION 2012, Stockholm, Sweden, June 14-15, 2012. Proceedings (Lecture Notes in Computer Science)*, Vol. 7274. Springer, 152–165.

Mieke Massink, Diego Latella, Andrea Bracciali, Michael D. Harrison, and Jane Hillston. 2012. Scalable context-dependent analysis of emergency egress models. *Formal Asp. Comput.* 24, 2 (2012), 267–302. DOI:http://dx.doi.org/10.1007/s00165-011-0188-1

C. McCaig. 2008. *From individuals to populations: changing scale in process algebra models of biological systems*. Ph.D. Dissertation. University of Stirling.

C. McCaig, R. Norman, and C. Shankland. 2008. Process Algebra Models of Population Dynamics. In *Proceedings of 3rd Int. Conference of Algebraic Biology (LNCS)*, Vol. 5147. Springer Verlag.

C. McCaig, R. Norman, and C. Shankland. 2009. From Individuals to Populations: A Symbolic Process Algebra Approach to Epidemiology. *Mathematics in Computer Science* 2, 3 (2009), 535–556.

Robin Milner. 1980. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science, Vol. 92. Springer. DOI:http://dx.doi.org/10.1007/3-540-10235-3

R. Norman and C. Shankland. 2003. Developing the Use of Process Algebra in the Derivation and Analysis of Mathematical Models of Infectious Disease. In *Proceedings of EUROCAST 2003 (LNCS)*, Vol. 2809. Springer Verlag.

A. Philippou, M. Toro, and M. Antonaki. 2013. Simulation and Verification in a Process Calculus for Spatially-Explicit Ecological Models. *Sci. Ann. Comp. Sci.* 23, 1 (2013), 119–167. DOI:http://dx.doi.org/10.7561/SACS.2013.1.119

C. Priami. 1995. Stochastic pi-Calculus. *Comput. J.* 38, 7 (1995), 578–589.

A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E.Y. Shapiro. 2004. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science* 325, 1 (2004), 141–167.

A. Regev and E. Shapiro. 2002. Cellular Abstractions: Cells as computation. *Nature* 419, 6905 (2002), 343–345.

M.L. Rettelbach and M. Siegle. 1994. Compositional Minimal Semantics for the Stochastic Process Algebra TIPP. In *Proc. of 2nd Process Algebra and Performance Modelling Workshop*, U. Herzog and M. Rettelbach (Eds.).

M. Silva and L. Recalde. 2002. Petri nets and integrality relaxations: A view of continuous Petri net models. *IEEE Trans. on Systems, Man, and Cybernetics, PartC* 32, 4 (2002), 314–327.

M. Silva and L. Recalde. 2004. On fluidification of Petri Nets: from discrete to hybrid and continuous models. *Annual Reviews in Control* 28, 2 (2004), 253–266.

B. Strulo. 1993. *Process Algebra for Discrete Event Simulation*. Ph.D. Dissertation. Imperial College.

D.T.J. Sumpter. 2000. *From Bee to Society: An Agent-based Investigation of Honey Bee Colonies*. Ph.D. Dissertation. University of Manchester.

C.M.N. Tofts. 1990. A Synchronous Calculus of Relative Frequency. In *CONCUR '90, Theories of Concurrency: Unification and Extension (LNCS)*, Vol. 458. Springer Verlag, 467–480.

M. Tribastone, S. Gilmore, and J. Hillston. 2012. Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.* 38, 1 (2012), 205–219.

K.S. Trivedi and V.G. Kulkarni. 1993. FSPNs: Fluid Stochastic Petri Nets. In *Application and Theory of Petri Nets*. LNCS, Vol. 691. Springer Verlag, 24–31.

L. Luisa Vissat, J. Hillston, G. Marion, and M.J. Smith. 2016. MELA: Modelling in Ecology with Location Attributes. In *Proceedings 14th International Workshop Quantitative Aspects of Programming Languages and Systems, QAPL'16 2016, Eindhoven, The Netherlands, April 2-3, 2016. (EPTCS)*, Vol. 227. 82–97.